*TLP* **9** (2): 213–238, 2009. © 2009 Cambridge University Press
doi:10.1017/S147106840900372X Printed in the United Kingdom

# *On finitely recursive programs*[1]

SABRINA BASELICE, PIERO A. BONATTI and GIOVANNI CRISCUOLO
*Università di Napoli "Federico II", Italy*
(*e-mail:* {baselice,bonatti,vanni}@na.infn.it)

*submitted 7 April 2008; revised 30 December 2008; accepted 16 January 2009*

**Abstract**

Disjunctive *finitary programs* are a class of logic programs admitting function symbols and hence infinite domains. They have very good computational properties; for example, ground queries are decidable, while in the general case the stable model semantics are $\Pi_1^1$-hard. In this paper we prove that a larger class of programs, called *finitely recursive programs*, preserve most of the good properties of finitary programs under the stable model semantics, which are as follows: (i) finitely recursive programs enjoy a compactness property; (ii) inconsistency checking and skeptical reasoning are semidecidable; (iii) skeptical resolution is complete for normal finitely recursive programs. Moreover, we show how to check inconsistency and answer skeptical queries using finite subsets of the ground program instantiation. We achieve this by extending the splitting sequence theorem by Lifschitz and Turner: we prove that if the input program $P$ is finitely recursive, then the partial stable models determined by any smooth splitting $\omega$-sequence converge to a stable model of $P$.

*KEYWORDS*: Answer set programming with infinite domains, infinite stable models, finitary programs, compactness, skeptical resolution.

## 1 Introduction

Answer set programming (ASP) (Marek and Truszczynski 1998; Niemelä 1999) is one of the most interesting achievements in the area of logic programming and nonmonotonic reasoning. It is a declarative problem-solving paradigm, mainly centered around some well-engineered implementations of the stable model semantics of logic programs (Gelfond and Lifschitz 1988, 1991), such as SMODELS (Niemelä and Simons 1997) and DLV (Eiter *et al.* 1997).

The most popular ASP languages are extensions of Datalog, namely, function-free, possibly disjunctive logic programs with negation as failure. The lack of function symbols has several drawbacks, related to expressiveness and encoding style (Bonatti 2004). In order to overcome such limitations and reduce the memory requirements of current implementations, a class of logic programs called *finitary programs* has been introduced (Bonatti 2004).

[1] This paper extends and refines Baselice *et al.* (2007).

In finitary programs function symbols (hence infinite domains) and recursion are allowed. However, recursion is restricted by requiring each ground atom to depend on finitely many ground atoms; such programs are called *finitely recursive*. Moreover, only finitely many ground atoms must occur in *odd cycles*—that is, cycles of recursive calls involving an odd number of negative subgoals—which means that there should be only finitely many potential sources of inconsistencies. These two restrictions bring a number of nice semantical and computational properties (Bonatti 2004). In general, function symbols make the stable model semantics highly undecidable (Marek and Remmel 2001). On the contrary, if the given program is finitary, then consistency checking, ground credulous queries, and ground skeptical queries are decidable. Nonground queries were proved to be r.e.-complete. Moreover, a form of compactness holds: an inconsistent finitary program has always a finite *unstable kernel*, i.e., a finite subset of the ground instantiation of the program with no stable models. All of these properties are quite unusual for a nonmonotonic logic.

As function symbols are being integrated in state-of-the-art reasoners such as DLV (Calimeri *et al*. 2008), it is interesting to extend these good properties to larger program classes. This goal requires a better understanding of the role of each restriction in the definition of finitary programs. It has already been noted (Bonatti 2004) that by dropping the first condition (i.e., if the program is not finitely recursive) one obtains a superclass of stratified programs, whose complexity is then far beyond computability. In the same paper, it is argued that the second restriction (on odd cycle) is needed for the decidability of ground queries. However, if a program is only finitely recursive (and infinitely many odd cycles are allowed), then the results of Bonatti (2004) do not characterize the exact complexity of reasoning and say nothing about compactness or about the completeness of the skeptical resolution calculus (Bonatti 2001b).

In this paper we extend and refine those results and prove that several important properties of finitary programs carry over to all disjunctive finitely recursive programs. We prove that for all such programs the compactness property still holds and that inconsistency checking and skeptical reasoning are semidecidable. Moreover, we extend the completeness of skeptical resolution (Bonatti 2001b, 2004) to all normal finitely recursive programs. Our results clarify the role that each of the two restrictions defining normal finitary programs has in ensuring their properties.

In order to prove these results we use program splittings (Lifschitz and Turner 1994), but the focus is shifted from splitting sequences (whose elements are sublanguages) to the corresponding sequences of subprograms that enjoy more invariant properties and may be regarded as a sort of normal form for splitting sequences. For this purpose we introduce the notion of *module sequence*. It turns out that disjunctive finitely recursive programs are exactly those disjunctive programs whose module sequences consist of finite elements. Moreover a disjunctive finitely recursive program $P$ has a stable model whenever each element $P_i$ of the sequence has a stable model, a condition which is not valid in general for all disjunctive programs with negation. This result provides an iterative characterization of the stable models of $P$. Module sequences and this theorem constitute a powerful formal tool that may facilitate the proof of new consistency results and provide a uniform framework for comparing different approaches to decidable reasoning with infinite domains.

The paper is organized as follows: The next section is devoted to preliminaries. In Section 3, we define module sequences and study their properties. In Section 4, we prove that every finitely recursive program with a consistent module sequence is consistent and use this result to extend the compactness property of finitary programs to all finitely recursive programs. Complexity results and two simple sound and complete algorithms for inconsistency checking and skeptical reasoning can be found in Section 5. Then, for a better, goal-directed calculus, the completeness theorem for skeptical resolution is extended to all finitely recursive programs in Section 6. Section 7 relates finitely recursive programs and our iterative approach to previous approaches to decidable reasoning with infinite stable models and makes a first step towards a unified picture based on our framework. Finally, Section 8 concludes the paper with a summary and a brief discussion of our results, as well as some interesting directions for future research.

## 2 Preliminaries

We assume the reader to be familiar with the classical theory of logic programming (Lloyd 1984).

*Disjunctive logic programs* are sets of (disjunctive) rules

$$A_1 \vee A_2 \vee \cdots \vee A_m \leftarrow L_1, \ldots, L_n \qquad (m > 0, n \geqslant 0),$$

where each $A_j$ $(j = 1, \ldots, m)$ is a logical atom and each $L_i$ $(i = 1, \ldots, n)$ is a *literal*, that is, either a logical atom $A$ or a negated atom $\mathtt{not}\,A$.

If $r$ is a rule with the above structure, then let $head(r) = \{A_1, A_2, \ldots, A_m\}$ and $body(r) = \{L_1, \ldots, L_n\}$. Moreover, let $body^+(r)$ (respectively $body^-(r)$) be the set of all atoms $A$ such that $A$ (respectively $\mathtt{not}\,A$) belongs to $body(r)$.

*Normal logic programs* are disjunctive logic programs whose rules $r$ have one atom in their head, that is, $|head(r)| = 1$.

The ground instantiation of a program $P$ is denoted by $\mathsf{Ground}(P)$, and the set of atoms occurring in $\mathsf{Ground}(P)$ is denoted by $atom(P)$. Similarly, $atom(r)$ denotes the set of atoms occurring in a ground rule $r$.

A Herbrand model $M$ of $P$ is a *stable model* of $P$ iff $M \in \mathsf{lm}(P^M)$, where $\mathsf{lm}(X)$ denotes the set of least models of a positive (possibly disjunctive) program $X$ and $P^M$ is the *Gelfond–Lifschitz transformation* (Gelfond and Lifschitz 1988, 1991) of $P$, obtained from $\mathsf{Ground}(P)$ by

(i) removing all rules $r$ such that $body^-(r) \cap M \neq \emptyset$ and
(ii) removing all negative literals from the body of the remaining rules.

Disjunctive and normal programs may have one, none, or multiple stable models. We say that a program is *consistent* if it has at least one stable model; otherwise the program is *inconsistent*. A *skeptical* consequence of a program $P$ is any closed first-order formula satisfied by all the stable models of $P$. A *credulous* consequence of $P$ is any closed first-order formula satisfied by at least one stable model of $P$.

The *dependency graph of a program* $P$ is a labeled directed graph, denoted by $DG(P)$, whose vertices are the ground atoms of $P$'s language. Moreover,

(i) there exists an edge labeled '+' (called positive edge) from $A$ to $B$ iff for some rule $r \in \mathsf{Ground}(P)$, $A \in head(r)$, and $B \in body(r)$;

(ii) there exists an edge labeled '-' (called negative edge) from $A$ to $B$ iff for some rule $r \in \mathsf{Ground}(P)$, $A \in head(r)$, and $\mathtt{not}\ B \in body(r)$;

(iii) there exists an unlabeled edge from $A$ to $B$ iff for some rule $r \in \mathsf{Ground}(P)$, $A \in head(r)$, and $B \in head(r)$.

An atom $A$ *depends positively* (respectively *negatively*) on $B$ if there is a directed path from $A$ to $B$ in the dependency graph with an even (respectively odd) number of negative edges. Moreover, each atom depends positively on itself. $A$ *depends* on $B$ if $A$ depends positively or negatively on $B$.

An *odd cycle* is a cycle in the dependency graph with an odd number of negative edges. A ground atom is *odd-cyclic* if it occurs in an odd cycle. Note that there exists an odd cycle iff some ground atom $A$ depends negatively on itself.

The class of programs on which this paper is focussed can now be defined very concisely.

*Definition 2.1*
A disjunctive program $P$ is *finitely recursive* iff each ground atom $A$ depends on finitely many ground atoms in $DG(P)$.[2]

For example, most standard list manipulation programs (like `member`, `append`, and `remove`) are finitely recursive. The reader can find numerous examples of finitely recursive programs in Bonatti (2004). In general, checking whether a program is finitely recursive is undecidable (Bonatti 2004). However, in Bonatti (2001a, 2004) a large decidable subclass has been implicitly characterized via static analysis techniques. Another expressive, decidable class of finitely recursive programs can be found in Simkus and Eiter (2007).

We will also mention frequently an important subclass of finitely recursive programs.

*Definition 2.2* (*Finitary programs*)
We say that a disjunctive program $P$ is *finitary* if the following conditions hold:

1. $P$ is finitely recursive;
2. there are finitely many odd-cyclic atoms in the dependency graph $DG(P)$.

Finitary programs have very good computational properties (for example, ground inferences are decidable). Many interesting programs, however, are finitely recursive but not finitary, due to integrity constraints that apply to infinitely many individuals.

*Example 2.3*
Typical programs for reasoning about actions and change are finitary. Figure 4 of Bonatti (2004) illustrates one of them, modeling a blocks world. That program defines—among others—two predicates $\mathtt{holds}(\mathit{fluent}, \mathit{time})$ and $\mathtt{do}(\mathit{action}, \mathit{time})$. The simplest way to add a constraint that forbids any parallel execution of two incompatible actions $a_1$ and $a_2$ is including a rule

$$f \leftarrow \mathtt{not}\, f, \mathtt{do}(a_1, T), \mathtt{do}(a_2, T)$$

[2] This definition differs from the one adopted in Bonatti (2002) because it is based on a different notion of dependency. Here the dependency graph contains edges between atoms occurring in the same head, while in Bonatti (2002) such dependencies are dealt with in a third condition in the definition of finitary programs. Further comparison with Bonatti (2002) can be found in Section 7.

in that program, where $f$ is a fresh propositional symbol (often such rules are equivalently expressed as *denials* like $\leftarrow \mathtt{do}(a_1, T), \mathtt{do}(a_2, T)$). This program is not finitary (because $f$ depends on infinitely many atoms since $T$ has an infinite range of values) but it can be reformulated as a finitely recursive program by replacing the above rule with

$$f(T) \leftarrow \mathtt{not}\, f(T), \mathtt{do}(a_1, T), \mathtt{do}(a_2, T)\,.$$

Note that the new program is finitely recursive but not finitary, because the new rule introduces infinitely many odd cycles (one for each instance of $f(T)$).

Our results on finitely recursive programs depend on the *splitting theorem* that allows to construct stable models in stages. In turn, this theorem is based on the notion of *splitting set*.

*Definition 2.4* (*Splitting set and bottom program; Lifschitz and Turner 1994; Baral 2003*)

A *splitting set* of a disjunctive logic program $P$ is any set $U$ of ground atoms such that, for all rules $r \in \mathsf{Ground}(P)$, if $head(r) \cap U \neq \emptyset$, then $atom(r) \subseteq U$. If $U$ is a splitting set for $P$, we also say that $U$ splits $P$. The set of rules $r \in \mathsf{Ground}(P)$ such that $head(r) \cap U \neq \emptyset$ is called the *bottom* of $P$ relative to the splitting set $U$ and is denoted by $bot_U(P)$. The subprogram $\mathsf{Ground}(P) \setminus bot_U(P)$ is called the *top* of $P$ relative to $U$.

The bottom program characterizes the restriction of the stable models of $P$ to the language determined by the splitting set. The top program determines the rest of each stable model; for this purpose it should be partially evaluated with respect to the stable models of the bottom.

*Definition 2.5* (*Partial evaluation; Lifschitz and Turner 1994; Baral 2003*)
The *partial evaluation* of a disjunctive logic program $P$ with splitting set $U$ with respect to a set of ground atoms $X$ is the program $e_U(\mathsf{Ground}(P), X)$ defined as follows:

$$\begin{aligned} e_U(\mathsf{Ground}(P), X) = \{ r' \mid & \text{ there exists } r \in \mathsf{Ground}(P) \text{ such that } (body^+(r) \cap U) \subseteq X, \\ & (body^-(r) \cap U) \cap X = \emptyset, head(r') = head(r), \\ & body^+(r') = body^+(r) \setminus U, \text{ and } body^-(r') = body^-(r) \setminus U \}\,. \end{aligned}$$

We are finally ready to formulate the splitting theorem (and hence the modular construction of stable models based on the top and bottom programs) in formal terms.

*Theorem 2.6* (*Splitting theorem; Lifschitz and Turner 1994*)
Let $U$ be a splitting set for a disjunctive logic program $P$. An interpretation $M$ is a stable model of $P$ iff $M = I \cup J$, where

1. $I$ is a stable model of $bot_U(P)$;
2. $J$ is a stable model of $e_U(\mathsf{Ground}(P) \setminus bot_U(P), I)$.

The splitting theorem has been extended to *transfinite sequences* in Lifschitz and Turner (1994). A (transfinite) sequence is a family whose index set is an initial segment of ordinals $\{\alpha \;:\; \alpha < \mu\}$. The ordinal $\mu$ is the *length* of the sequence.

A sequence $\langle U_\alpha \rangle_{\alpha<\mu}$ of sets is *monotone* if $U_\alpha \subseteq U_\beta$ whenever $\alpha < \beta$ and *continuous* if for each limit ordinal $\alpha < \mu$, $U_\alpha = \bigcup_{\nu<\alpha} U_\nu$.

*Definition 2.7* (*Splitting sequence, Lifschitz and Turner 1994*)
A *splitting sequence* for a disjunctive program $P$ is a monotone, continuous sequence $\langle U_\alpha \rangle_{\alpha<\mu}$ of splitting sets for $P$ such that $\bigcup_{\alpha<\mu} U_\alpha = atom(\mathsf{Ground}(P))$.

Lifschitz and Turner (1994) generalize the splitting theorem to splitting sequences.

*Theorem 2.8* (*Splitting sequence theorem; Lifschitz and Turner 1994*)
Let $P$ be a disjunctive program;[3] $M$ is a stable model of $P$ iff there exists a splitting sequence $\langle U_\alpha \rangle_{\alpha<\mu}$ such that

1. $M_0$ is a stable model of $bot_{U_0}(P)$;
2. for all successor ordinals $\alpha < \mu$, $M_\alpha$ is a stable model of $e_{U_{\alpha-1}}(bot_{U_\alpha}(P) \setminus bot_{U_{\alpha-1}}(P), \bigcup_{\beta<\alpha} M_\beta)$;
3. for all limit ordinals $\lambda < \mu$, $M_\lambda = \emptyset$;
4. $M = \bigcup_{\alpha<\mu} U_\alpha$.

## 3 Module sequences and a normal form for splitting sequences

In this section we replace the sequences of program slices $bot_{U_\alpha}(P) \setminus bot_{U_{\alpha-1}}(P)$ adopted by Lifschitz and Turner (1994) with slightly different and simpler program module sequences. Then we prove some properties of module sequences that will be useful in proving our main results.

*Definition 3.1* (*GH, Module sequence*)
Let $P$ be a disjunctive program, and let the set of its *ground head atoms* be

$$GH = \{\, p \mid p \in head(r),\ r \in \mathsf{Ground}(P) \,\}.$$

The *module sequence* $P_1, P_2, \ldots, P_n, \ldots$ *induced by an enumeration* $p_1, p_2, \ldots, p_n, \ldots$ *of* $GH$ is defined as follows:

$$\begin{aligned} P_1 &= \{\, r \in \mathsf{Ground}(P) \mid p_1 \text{ depends on some } A \in head(r) \,\}, \\ P_{i+1} &= P_i \cup \{\, r \in \mathsf{Ground}(P) \mid p_{i+1} \text{ depends on some } A \in head(r) \,\} \qquad (i \geqslant 1). \end{aligned}$$

Of course, we are particularly interested in those properties of module sequences that are independent from the enumeration of $GH$. We say that a ground subprogram $P' \subseteq \mathsf{Ground}(P)$ is *downward closed* if for each atom $A$ occurring in $atom(P')$ the subprogram $P'$ contains all the rules $r \in \mathsf{Ground}(P)$ such that $A \in head(r)$.

*Proposition 3.2*
Let $P$ be a disjunctive program. For all module sequences $P_1, P_2, \ldots,$ for $P$:

1. $\bigcup_{i\geqslant 1} P_i = \mathsf{Ground}(P)$;
2. for each $i \geqslant 1$ and $j \geqslant i$, $atom(P_i)$ is a splitting set of $P_j$, and $P_i = bot_{atom(P_i)}(P_j)$;
3. for each $i \geqslant 1$, $atom(P_i)$ is a splitting set of $P$, and $P_i = bot_{atom(P_i)}(P)$;
4. for each $i \geqslant 1$, $P_i$ is downward closed.

[3] The splitting sequence theorem holds for disjunctive logic programs extended with so-called strong negation that, however, is essentially syntactic sugar. Therefore, for the sake of simplicity, we ignore it here.

This proposition follows easily from the definitions. It shows that each module sequence for $P$ consists of the bottom programs corresponding to a particular splitting sequence $\langle atom(P_i)\rangle_{i<\omega}$ that depends on the underlying enumeration of $GH$. Roughly speaking, such sequences (whose lengths are limited by $\omega$) constitute a *normal form* for splitting sequences and enjoy useful properties that are invariant with respect to the enumeration.

*Definition 3.3* (*Smoothness*)
A transfinite sequence of sets $\langle X_\alpha\rangle_{\alpha<\mu}$ is *smooth* iff $X_0$ is finite, and for each non-limit ordinal $\alpha + 1 < \mu$, the difference $X_{\alpha+1} \setminus X_\alpha$ is finite.

Note that when $\mu = \omega$ (as in module sequences), smoothness implies that each $X_\alpha$ in the sequence is finite. Finitely recursive programs are completely characterized by smooth module sequences.

*Theorem 3.4*
For all disjunctive logic programs $P$, the following are equivalent:

1. $P$ is finitely recursive;
2. $P$ has a smooth module sequence (where each $P_i$ is finite);
3. all module sequences for $P$ are smooth.

*Proof*
(1 ⇒ 3) Let $P$ be a finitely recursive program, and let $e = p_1, p_2, \ldots$ be any enumeration of $GH$. If $S = P_1, P_2, \ldots$ is the module sequence induced by the enumeration $e$, then $S$ is smooth because, for each atom $p_i$ in $e$, the set $\{\, r \in \mathsf{Ground}(P) \mid p_i \textit{ depends on some } A \in \textit{head}(r) \,\}$ is finite, as $P$ is finitely recursive. Since this holds for an arbitrary enumeration $e$, all module sequences for $P$ are smooth.
(3 ⇒ 2) Trivial.
(2 ⇒ 1) Let $S = P_1, P_2, \ldots,$ be a smooth module sequence for $P$, and let $p$ be an atom in $\mathsf{Ground}(P)$. By Proposition 3.2.(1), there is a program $P_i$ in $S$ such that $p \in atom(P_i)$. Moreover, $P_i$ is downward closed by definition of module sequence, and it is finite because $S$ is smooth. Then $p$ depends only on finitely many ground atoms. Since $p$ has been arbitrarily chosen, the same holds for all ground atoms; therefore $P$ is finitely recursive. □

Smooth module sequences clearly correspond to smooth splitting sequences of length $\omega$. In particular, for each smooth module sequence $\langle P_i\rangle_{i<\omega}$ , $\langle atom(P_i)\rangle_{i<\omega}$ is a smooth splitting sequence. Conversely, given a smooth splitting sequence $\langle U_i\rangle_{i<\omega}$ and an arbitrary enumeration $p_1, p_2, \ldots, p_i$, the resulting module sequence must necessarily be smooth. Suppose it is not; then some $p_i$ must depend on infinitely many atoms. Consequently, all the sets $U_j$ containing $p_i$ should be infinite as well (a contradiction). Note that in general a smooth splitting sequence does not strictly correspond to a module sequence. For example, the difference between two consecutive elements of a splitting sequence may contain two atoms that do not depend on each other, while this is impossible in module sequences by construction.

Using the above relationships between smooth module sequences and smooth splitting sequences of length $\omega$, the characterization of finitely recursive programs can be completed as follows, in terms of standard splitting sequences:

*Corollary 3.5*
For all disjunctive programs $P$, the following are equivalent:

1. $P$ is finitely recursive;
2. $P$ has a smooth splitting sequence of length $\mu \leqslant \omega$.

*Proof*
A straightforward consequence of Theorem 3.4 and the correspondence between smooth module sequences and smooth splitting sequences of length $\mu \leqslant \omega$. $\square$

Note the asymmetry between Corollary 3.5 and Theorem 3.4. It can be explained by the generality of splitting sequences: even if the underlying program is finitely recursive, splitting sequences are not forced to be all smooth. For example, the finitely recursive program

$$\begin{array}{rcl} even(0) & & \\ even(s(s(X))) & \leftarrow & even(X) \\ odd(s(0)) & & \\ odd(s(s(X))) & \leftarrow & odd(X) \end{array}$$

has a non-smooth splitting sequence $\langle \{even(s^n(X)) \mid n \text{ even}\}, \{odd(s^n(X)) \mid n \text{ odd}\} \rangle$ .

Next we illustrate how module sequences provide an incremental characterization of the stable models of disjunctive logic programs.

Roughly speaking, the following theorem rephrases the splitting sequence theorem of Lifschitz and Turner (1994) in terms of module sequences. The original splitting sequence theorem applies to sequences of disjoint program "slices," while our theorem applies to monotonically increasing program sequences. Since no direct proof of the splitting sequence theorem was ever published (only the proof of a more general result for default logic was published; Tuner 1996), here we give a direct proof of our result.

*Theorem 3.6* (*Module sequence theorem*)
Let $P$ be a disjunctive program and $P_1, P_2, \ldots$ be a module sequence for $P$. Then $M$ is a stable model of $P$ iff there exists a sequence $M_1, M_2, \ldots$ such that:

1. for each $i \geqslant 1$, $M_i$ is a stable model of $P_i$;
2. for each $i \geqslant 1$, $M_i = M_{i+1} \cap atom(P_i)$;
3. $M = \bigcup_{i \geqslant 1} M_i$.

*Proof*
Let $M$ be a stable model of $P$. Since $P_1, P_2, \ldots$ is a module sequence for $P$, we have that for each $i \geqslant 1$, $atom(P_i)$ is a splitting set of $P$ and $P_i = bot_{atom(P_i)}(P)$. Consider the sequence of models $M_i = M \cap atom(P_i)$, $1 \leqslant i < \omega$. By the splitting theorem (Lifschitz and Turner 1994), for each $i \geqslant 1$, $M_i$ is a stable model of $P_i$. Second, since $P_{i+1} \supseteq P_i$, we have $M_i = M \cap atom(P_i) = (M \cap atom(P_{i+1})) \cap atom(P_i) = M_{i+1} \cap atom(P_i)$. Finally, by Proposition 3.2.(1) we have $\bigcup_i M_i = M$. Then for each stable model $M$ of $P$ there exists a sequence of finite sets of ground atoms that satisfies properties 1, 2, and 3.

Conversely, let $P$ be a disjunctive logic program. For the sake of readability, we assume without loss of generality that $P$ is ground. Suppose that there exists a sequence

$M_1, M_2, \dots$ that satisfies properties 1, 2, and 3. We have to prove that the set $M = \bigcup_{i \geqslant 1} M_i$ is a stable model of $P$; equivalently,

$$\bigcup_{i \geqslant 1} M_i \in \mathsf{lm}(P^M).$$

Properties 2 and 3 imply that for all $i \geqslant 1$, $(M \cap atom(P_i)) = M_i$; consequently $P_i^M = P_i^{M_i}$, and by Proposition 3.2.(1),

$$P^M = \left( \bigcup_{i \geqslant 1} P_i \right)^M = \bigcup_{i \geqslant 1} P_i^M = \bigcup_{i \geqslant 1} P_i^{M_i}. \tag{1}$$

First we prove that $M$ is a model of $P^M$; that is, for each rule $r$ in $P^M$, if $body(r) \subseteq M$, then $head(r) \cap M \neq \emptyset$. Let $r$ be any rule in $P^M$ such that $body(r) \subseteq M$. By equation (1), there is an integer $i \geqslant 1$ such that $r \in P_i^{M_i}$. Moreover, it is not hard to prove that properties 2 and 3 and $body(r) \subseteq M$ imply $body(r) \subseteq M_i$. Now, since $M_i$ is a stable model of $P_i$ and $body(r) \subseteq M_i$, we have $head(r) \cap M_i \neq \emptyset$. It follows immediately that $head(r) \cap M \neq \emptyset$. Since this holds for any $r \in P^M$, we conclude that $M$ is a model of $P^M$.

We are left to show that $M$ is a minimal model for $P^M$. Suppose that $P^M$ has a model $M' \subset M$. Let $p \in (M \setminus M')$, and let $i$ be an integer such that $p \in atom(P_i)$. Since $P_i^M = P_i^{M_i}$ is a bottom program for $P^M$, $M' \cap atom(P_i)$ is a model for $P_i^{M_i}$ and is strictly contained in $M_i$, but this is a contradiction because by hypothesis $M_i$ is a minimal model of $P_i^{M_i}$. □

The module sequence theorem (respectively, the splitting sequence theorem) suggests a relationship between the consistency of a program $P$ and the consistency of each step in $P$'s module sequences (respectively, the sequence of program slices induced by $P$'s splitting sequences). To clarify this point we introduce another invariant property of module sequences.

*Definition 3.7*
A module sequence $S = P_1, P_2, \dots$ for a disjunctive program $P$ is *inconsistent* if, for some $i < \omega$, $P_i$ is inconsistent; otherwise $S$ is *consistent*.

*Proposition 3.8*
If a disjunctive program $P$ has an inconsistent module sequence, then $P$ is inconsistent.

*Proof*
Suppose that $P$ has an inconsistent module sequence $P_1, P_2, \dots$; that is, some $P_i$ in the sequence is inconsistent. It follows that $P$ has an inconsistent bottom program, and hence $P$ is inconsistent by the splitting theorem. □

*Theorem 3.9*
Let $S = P_1, P_2, \dots$ be a module sequence for a disjunctive program $P$. If $S$ is inconsistent, then each module sequence for $P$ is inconsistent.

*Proof*
Let $S = P_1, P_2, \dots$ be an inconsistent module sequence for $P$ induced by the enumeration $p_1, p_2, \dots$ of $GH$, and let $i$ be the least index such that $P_i$ is inconsistent. Let $S' = P'_1, P'_2, \dots$ be any module sequence for $P$ induced by the enumeration $p'_1, p'_2, \dots$ of $GH$. Since $i$ is

finite, there exists a finite $k$ such that $\{p_1, p_2, \ldots, p_i\} \subseteq \{p'_1, p'_2, \ldots, p'_k\}$. So, by construction, $P_i \subseteq P'_k$, and then $atom(P_i) \subseteq atom(P'_k)$. Moreover, by definition, $P_i$ is downward closed; therefore $P_i = bot_{atom(P_i)}(P'_k)$. Since $P_i$ is inconsistent, $P'_k$ is inconsistent (by the splitting theorem), and hence $S'$ is inconsistent, too. □

In other words, for a given program $P$, all module sequences are either inconsistent or consistent. In particular, if $P$ is consistent, then every member $P_i$ of any module sequence for $P$ must be consistent. The converse property would allow to define a procedure for enumerating the stable models of $P$ (as shown in the following sections). Unfortunately, even if each step in a module sequence is consistent, the entire program $P$ is not necessarily consistent, as shown by the following example:

*Example 3.10*
As a preliminary step, consider the following program $P_f$ (due to Fages 1994):

$$\begin{aligned}&q(X) \leftarrow q(f(X)).\\ &q(X) \leftarrow \texttt{not}\, q(f(X)).\\ &r(0).\end{aligned}$$

The third rule is only needed to introduce the constant 0 into the program's language.

This program is inconsistent. To see this, note that—roughly speaking—the first two rules in $P_f$ are classically equivalent to

$$q(X) \leftarrow q(f(X)) \vee \texttt{not}\, q(f(X))\,.$$

Since the body is a tautology, and the stable models of a program are also classical models of the program (if `not` is interpreted as $\neg$), we have that every stable model of $P_f$ should satisfy all ground instances of $q(X)$. However, the Gelfond–Lifschitz transformation with respect to such a model would contain only the first and the third program rules, and hence the least model of the transformation would contain no instance of $q(X)$. It follows that $P_f$ is inconsistent (it has no stable models). Now consider the following extension $P$ of $P_f$:

1. $q(X) \leftarrow q(f(X)), p(X)$.
2. $q(X) \leftarrow \texttt{not}\, q(f(X)), p(X)$.
3. $r(0)$.
4. $p(X) \leftarrow \texttt{not}\, p'(X)$.
5. $p'(X) \leftarrow \texttt{not}\, p(X)$.
6. $c(X) \leftarrow \texttt{not}\, c(X), \texttt{not}\, p(X)$.

The program $P$ is inconsistent, too. To verify it, suppose that $M$ is a stable model of $P$. By rules 4 and 5, for all ground instances of $X$, exactly one of $p(X)$ and $p'(X)$ is true in $M$. However, if $p(X)$ is false, then rule 6 produces an inconsistency due to the odd cycle involving $c(X)$. It follows that all ground instances of $p(X)$ must be true in $M$. But then rules 1, 2 and 3 become equivalent to program $P_f$ and prevent $M$ from being a stable model, as explained above. So $P$ is inconsistent.

Next, consider the enumeration $e = r(0), q(0), p(0), p'(0), c(0), q(f(0)), p(f(0)), p'(f(0)), c(f(0)), \ldots,$ of the set $GH$. This enumeration induces the following module sequence for $P$ (where the expression $[X/t]$ denotes the substitution mapping $X$ onto $t$):

$$
\begin{aligned}
P_0 &= \{r(0)\},\\
P_1 &= P_0 \cup \bigcup_{k<\omega} \{ q(X) \leftarrow q(f(X)), p(X),\\
&\qquad\qquad q(X) \leftarrow \texttt{not}\, q(f(X)), p(X),\\
&\qquad\qquad p(X) \leftarrow \texttt{not}\, p'(X),\\
&\qquad\qquad p'(X) \leftarrow \texttt{not}\, p(X) \,\}\, [X/f^k(0)],\\
P_{i+1} &= P_i \cup \{c(X) \leftarrow \texttt{not}\, c(X), \texttt{not}\, p(X)\}\, [X/f^{i-1}(0)] \qquad (i \geqslant 1)\,.
\end{aligned}
$$

Note that $M_0 = \{r(0)\}$ is a stable model of $P_0$, and for each $i \geqslant 1$ and $k \geqslant i-2$

$$
\begin{aligned}
M_i^k = \{&r(0), p(f^0(0)), p(f^1(0)), p(f^2(0)), \dots, p(f^k(0)),\\
&p'(f^{k+1}(0)), p'(f^{k+2}(0)), \dots, p'(f^{k+j}(0)), \dots\\
&q(f^0(0)), q(f^1(0)), q(f^2(0)), \dots, q(f^k(0))\}
\end{aligned}
$$

is a stable model of $P_i$. Therefore, each $P_i$ is consistent, while $\bigcup_i P_i = \mathsf{Ground}(P)$ is inconsistent. This happens because for each stable model $M$ of $P_1$ there exists a $P_j$ $(j > 1)$ such that $M$ is not the bottom part of any stable model of $P_j$. Intuitively, $M$ has been "eliminated" at step $j$. In this example $P_1$ has infinitely many stable models, and it turns out that no finite step eliminates them all. Consequently, each $P_i$ in the module sequence is consistent, but the entire program is not.

Note that $P$ is not finitely recursive because, for each grounding substitution $\sigma$, $q(X)\sigma$ depends on the infinite set of ground atoms $\{\, q(f(X))\sigma,\ q(f(f(X)))\sigma, \dots \}$ (due to rules 1 and 2). In the following section we are going to prove that finitely recursive programs are not affected by the problem illustrated in Example 3.10; that is, they enjoy the converse of Theorem 3.9. This property will be used to design an effective enumeration procedure for their stable models.

## 4 Compactness property of disjunctive finitely recursive programs

Here we prove that the compactness theorem proved in Baselice *et al.* (2007) for normal finitely recursive programs actually holds for all disjunctive finitely recursive programs.

The first step is to prove the converse of Theorem 3.9 for all finitely recursive programs.

*Theorem 4.1*
For all disjunctive finitely recursive programs $P$, if some module sequence for $P$ is consistent, then $P$ is consistent.

*Proof*
Let $S$ be any module sequence for $P$. If $S$ is consistent, then each module $P_i$ in $S$ has a nonempty set of stable models. It suffices to prove that there exists a sequence $M_1, M_2, \dots$ of stable models of $P_1, P_2, \dots$, respectively, that satisfy the properties of Theorem 3.6, because this implies that $M = \bigcup_i M_i$ is a stable model of $P$.

We call a stable model $M_i$ of $P_i$ *bad* if there exists a $k > i$ such that no stable model $M_k$ of $P_k$ extends $M_i$; otherwise it is called *good*. We say that $M_k$ extends $M_i$ if $M_k \cap atom(P_i) = M_i$.

*Claim 1*: Each $P_i$ must have at least one *good* stable model.

To prove the claim, suppose that there exists an $i$ such that all stable models of $P_i$ are *bad*. Since $P_i$ is a finite program, it has a finite number $M_{i,1}, \dots, M_{i,r}$ of stable models. By assumption, for each $M_{i,j}$ there is a program $P_{k_j}$ none of whose stable models extends $M_{i,j}$. Let $k = max\{k_1, \dots, k_r\}$; clearly, no stable model of $P_k$ extends any stable model of $P_i$, and this is a contradiction because $P_k$, by hypotheses, has at least one stable model $M_k$, and by the splitting theorem, $M_k$ must extend a stable model of $P_i$. This proves Claim 1.

*Claim 2*: Each *good* stable model $M_i$ of $P_i$ is extended by some *good* stable model $M_{i+1}$ of $P_{i+1}$.

Suppose it is not. Then, none of the stable models $M_{i+1,1}, \dots, M_{i+1,r}$ of $P_{i+1}$ that extend the good stable model $M_i$ of $P_i$ is good. This implies (by analogy with the proof of Claim 1) that there exists a module $P_k$ $(k > i + 1)$ none of whose stable models extends any of $M_{i+1,1}, \dots, M_{i+1,r}$. It follows that none of $P_k$'s stable models can extend $M_i$, and this contradicts the hypothesis that $M_i$ is good.

From the two claims it follows immediately that there exists an infinite sequence $M_1$, $M_2, \dots$ that satisfies properties 1 and 2 of Theorem 3.6, and hence the union $M = \bigcup_i M_i$ is a stable model of $P$. $\square$

Note that in Example 3.10, module $P_1$ is infinite and has infinitely many stable models, all of which are "bad". Each of them is eliminated at some step, but no finite step eliminates them all, which is why that module sequence is consistent, although the entire program $P$ is not.

Theorem 4.1 can be extended to all smooth splitting sequences with length $\omega$.

*Corollary 4.2*
Let $\langle U_\alpha \rangle_{\alpha<\omega}$ be a smooth splitting sequence for a disjunctive program $P$. Then $P$ is consistent iff for all $\alpha < \omega$, $bot_{U_\alpha}(P)$ is consistent.

*Proof*
A straightforward consequence of the correspondence between module and splitting sequences. $\square$

The restriction to sequences with length $\omega$ is essential to derive the above corollary, which is not valid otherwise, as shown by the following example:

*Example 4.3*
Let $P$ be the following program, where rule 1 has the role of creating an infinite Herbrand domain:

1. $r(f(0))$.
2. $p(X) \leftarrow \texttt{not}\, q(X)$.
3. $q(X) \leftarrow \texttt{not}\, p(X)$.
4. $some\_q \leftarrow q(X)$.
5. $f \leftarrow \texttt{not}\, f,\ \texttt{not}\, some\_q$.
6. $c(X) \leftarrow \texttt{not}\, c(X), q(X)$.

This program is inconsistent for the following reasons: For all ground instances of $X$, rules 2 and 3 force exactly one of $p(X)$ and $q(X)$ to be true. If no instance of $q(X)$ is true, then rules 4 and 5 create a contradiction by "activating" the odd cycle involving

$f$. However, if some instance of $q(X)$ is true, then rule 6 generates a contradiction by "activating" the odd cycle involving $c(X)$. It follows that $P$ has no stable models.

However, $P$ has a smooth splitting sequence with length $2\omega$ whose bottom programs are all consistent:

$$\begin{aligned}
U_0 &= \{\, p(0),\ q(0),\ r(0) \,\}, \\
U_i &= U_{i-1} \cup \{\, p(f^i(0)),\ q(f^i(0)),\ r(f^i(0)) \,\} && (0 < i < \omega), \\
U_\omega &= \bigcup_{i<\omega} U_i, \\
U_{\omega+1} &= U_\omega \cup \{\, \mathit{some\_q},\ f,\ c(0) \,\}, \\
U_{\omega+j+1} &= U_{\omega+j} \cup \{\, c(f^j(0)) \,\} && (0 \leqslant j < \omega).
\end{aligned}$$

In particular,

(i) $bot_{U_\omega}(P)$ has infinitely many stable models, one for each choice between $p(X)$ and $q(X)$, for all instances of $X$;
(ii) $bot_{U_{\omega+1}}(P)$ keeps all the stable models where at least one instance of $q(X)$ is true;
(iii) $bot_{U_{\omega+j+1}}(P)$ keeps only those stable models where the first true instance of $q(X)$ is $q(f^k(0))$ with $k > j$.

Now we are ready to extend the compactness property of finitary normal programs to all disjunctive finitely recursive programs.

*Definition 4.4*
An *unstable kernel* for a disjunctive program $P$ is a set $K \subseteq \mathsf{Ground}(P)$ with the following properties:

1. $K$ is downward closed;
2. $K$ has no stable model.

*Theorem 4.5* (*Compactness*)
A disjunctive finitely recursive program $P$ has no stable model iff it has a finite unstable kernel.

*Proof*
By Proposition 3.8 and Theorem 4.1, $P$ has no stable model iff it has an inconsistent module sequence. So, let $P_1, P_2, \ldots, P_n, \ldots$ be an inconsistent module sequence for $P$ and choose an index $i \geqslant 1$ such that $P_i$ is inconsistent. By Proposition 3.2, $P_i \subseteq \mathsf{Ground}(P)$; moreover, $P_i$ is downward closed. Then $P_i$ is an unstable kernel for $P$. Moreover, by Theorem 3.4, $P_i$ is finite. □

## 5 Reasoning with disjunctive finitely recursive programs

By taking an effective enumeration of the set $GH$ of ground head atoms, one can effectively compute each element of the corresponding module sequence. Let us call CONSTRUCT$(P, i)$ an effective procedure that, given a finitely recursive program $P$ and an index $i$, returns the ground program $P_i$, and let $SM\,(P_i)$ be an algorithm that computes the finite set of the finite stable models of $P_i$:

*Theorem 5.1*
Let $P$ be a disjunctive finitely recursive program. Whether $P$ is inconsistent is semidecidable.

*Proof*
Given a module sequence $P_1, P_2, \ldots, P_n, \ldots$ for the program $P$, consider the algorithm CONSISTENT $(P)$.

**Algorithm** CONSISTENT $(P)$
```
1: $i = 0$;
2: $answer$ = TRUE;
3: repeat
4:     $i = i + 1$;
5:     $P_i$ =CONSTRUCT $(P, i)$;
6:     if $SM(P_i) = \emptyset$ then
7:         $answer$ = FALSE;
8: until $\neg answer$ OR $P_i$ = Ground($P$)
9: return $answer$;
```

By Proposition 3.8 and Theorem 4.1, $P$ is inconsistent iff there exists an $i \geqslant 1$ such that $P_i$ is inconsistent (note that we can always check the consistency of $P_i$ because $P_i$ is finite). Then, the algorithm returns FALSE iff $P$ is inconsistent.

Note that if Ground($P$) is infinite, then any module sequence for $P$ is infinite, and the algorithm CONSISTENT $(P)$ terminates iff $P$ is not consistent. □

Next we deal with skeptical inference. Recall that a closed first-order formula $F$ is a skeptical consequence of $P$ iff $F$ is satisfied (according to classical semantics) by all the stable models of $P$.

*Theorem 5.2*
Let $P$ be a disjunctive finitely recursive program and $P_1$, $P_2, \ldots$ be a module sequence for $P$. A ground formula $F$ in the language of $P$ is a skeptical consequence of $P$ iff there exists a finite $k \geqslant 1$ such that $F$ is a skeptical consequence of $P_k$ and $atom(F) \subseteq atom(P_k)$.

*Proof*
Let $h$ be the least integer such that $atom(F) \subseteq atom(P_h)$. (Note that there always exists such an $h$ because $atom(F)$ is finite.) Suppose that there exists a $k \geqslant h$ such that $F$ is a skeptical consequence of $P_k$. Since $P_k$ is a bottom program for $P$, each stable model $M$ of $P$ extends a stable model $M_k$ of $P_k$ and then satisfies $F$. (Here the assumption that $atom(F) \subseteq atom(P_k)$ is essential to conclude that $M$ and $M_k$ agree on the truth of $F$.) So, $F$ is a skeptical consequence of $P$. This proves the "if" part.

Now suppose that, for each $k \geqslant h$, $F$ is not a skeptical consequence of $P_k$. This implies that each $P_k$ is consistent (hence $P$ is consistent), and, moreover, the set $S$ of all the stable models of $P_k$ that falsify $F$ is not empty.

Note that $S$ is finite because $P_k$ is finite (as $P$ is finitely recursive). So, if all the models in $S$ are *bad* (cf. the proof of Theorem 4.1), then there exists a finite integer $j > k$ such that no model of $P_j$ contains any model of $S$. Consequently, $F$ is a skeptical consequence of $P_j$—a contradiction.

Therefore at least one of these models must be *good*. Then there must be a model $M$ of $P$ that contains this *good* model of $P_k$, and hence $F$ is not a skeptical consequence of $P$. □

The next theorem follows easily.

*Theorem 5.3*
Let $P$ be a disjunctive finitely recursive program. For all ground formulas $F$, the problem of deciding whether $F$ is a skeptical consequence of $P$ is semidecidable.

*Proof*
Given a module sequence $P_1, P_2, \ldots, P_n, \ldots$ for the program $P$, consider the algorithm SKEPTICAL $(P, F)$.

**Algorithm** SKEPTICAL $(P, F)$

```
1:  $answer$ = FALSE;
2:  $i = 0$;
3:  repeat
4:      $i = i + 1$;
5:      $P_i$ = CONSTRUCT $(P, i)$;
6:  until $atom(F) \subseteq atom(P_i)$
7:  repeat
8:      if $SM(P_i) = \emptyset$ OR $P_i$ skeptically entails $F$ then
9:          $answer$ = TRUE;
10:     else
11:         $i = i + 1$;
12:         $P_i$ = CONSTRUCT $(P, i)$;
13: until $answer$ OR $P_i = P$
14: return $answer$;
```

For each $P_i$ such that $atom(F) \subseteq atom(P_i)$, the algorithm SKEPTICAL $(P, F)$ checks if $F$ is a skeptical consequence of $P_i$. Since $P_i$ is finite, we can always decide if $F$ is a skeptical consequence of $P_i$. So, by Theorem 5.2, the algorithm returns TRUE iff $F$ is a skeptical consequence of $P$.

Note that if Ground($P$) is infinite, then any module sequence for $P$ is infinite, and the algorithm SKEPTICAL $(P, F)$ teminates iff $F$ is a skeptical consequence of $P$. □

For a complete characterization of the complexity of ground queries and inconsistency checking, we are only left to prove that the above upper bounds are tight. Actually, we prove slightly stronger lower bounds that hold even for *normal* finitely recursive programs.

*Theorem 5.4*
Deciding whether a normal finitely recursive program $P$ is inconsistent is r.e.-hard.

*Proof*
The proof is by reduction of the problem of skeptical inference of a quantified formula over a finitary normal program (proved to be r.e.-complete in Bonatti 2004, Corollary 23) to the problem of inconsistency checking over a normal finitely recursive program.

Let $P$ be a finitary program and $\exists F$ be a closed existentially quantified formula. Let $((l_{11} \vee l_{12} \vee \dots) \wedge (l_{21} \vee l_{22} \vee \dots) \wedge \dots)$ be the conjunctive normal form of $\neg F$. Then $\exists F$ is a skeptical consequence of $P$ iff the program $P \cup C$ is inconsistent, where

$$C = \left\{ \begin{array}{c} p_1(\vec{x}_1) \leftarrow \mathtt{not}\, l_{11}, \mathtt{not}\, l_{12}, \dots, \mathtt{not}\, p_1(\vec{x}_1) \\ p_2(\vec{x}_2) \leftarrow \mathtt{not}\, l_{21}, \mathtt{not}\, l_{22}, \dots, \mathtt{not}\, p_2(\vec{x}_2) \\ \vdots \end{array} \right\},$$

$p_1, p_2, \dots$ are new atom symbols not occurring in $P$ or $F$, and $\vec{x}_i$ is the vector of all variables occurring in $(l_{i1} \vee l_{i2} \vee \dots)$. Note that $P \cup C$ is a normal finitely recursive program.

The constraints in $C$ add no model to $P$, but they only discard those models of $P$ that satisfy $F\theta$ (for some substitution $\theta$). So let $SM(P)$ be the set of stable models of $P$. Then each model in $SM(P \cup C)$ satisfies $\forall\neg F$. Iff either $SM(P) = \emptyset$ or all stable models of $P$ satisfy $\exists F$, $SM(P \cup C) = \emptyset$ (that is, $P \cup C$ is inconsistent). Then $SM(P \cup C) = \emptyset$ iff $\exists F$ is a skeptical consequence of $P$. □

*Theorem 5.5*
Deciding whether a normal finitely recursive program $P$ skeptically entails a ground formula $F$ is r.e.-hard.

*Proof*
The proof is by reduction of inconsistency checking for normal finitely recursive programs to the problem of skeptical inference of a ground formula from a normal finitely recursive program.

Let $P$ be a normal finitely recursive program and $q$ be a new ground atom that doesn't occur in $P$. Then, $P$ is inconsistent iff $q$ is a skeptical consequence of $P$. Since $q$ occurs in the head of no rule of $P$, $q$ cannot occur in a model of $P$. So, $P$ skeptically entails $q$ iff $P$ has no stable model. □

*Corollary 5.6*
Deciding whether a disjunctive finitely recursive program $P$ credulously entails a ground formula $F$ is co-r.e.-complete.

*Proof*
The proof follows immediately from Theorems 5.3 and 5.5 and from the fact that a ground formula $F$ is a credulous consequence of $P$ iff $\neg F$ is not a skeptical consequence of $P$. □

## 6 Skeptical resolution for finitely recursive normal programs

In this section we extend the work in Bonatti (2001b, 2004) by proving that skeptical resolution (a top-down calculus which is known to be complete for Datalog and normal finitary programs under the skeptical stable model semantics) is complete also for the class of finitely recursive normal programs. Skeptical resolution has several interesting properties. For example, it does not require the input program $P$ to be instantiated before reasoning (unlike the major state-of-the-art stable model reasoners), and it can produce nonground (i.e., universally quantified) answer substitutions. The goal-directed nature of skeptical resolution makes it more interesting than the naive algorithms illustrated in Section 5.

We are not describing all the formal details of the calculus here—the reader is referred to Bonatti (2001b). Skeptical resolution is based on *goals with hypotheses* (*h-goals* for short) which are pairs $(G \mid H)$, where $H$ and $G$ are finite sequences of literals. Roughly speaking, the answer to a query $(G \mid H)$ should be *yes* if $G$ holds in all the stable models that satisfy $H$. Hence $(G \mid H)$ has the same meaning in answer set semantics as the implication $(\bigwedge G \leftarrow \bigwedge H)$. Finally, a *skeptical goal* (*s-goal* for short) is a finite sequence of h-goals.

The calculus consists of five inference rules given next.

## *6.1 Resolution*

This rule may take two forms; a literal can be unified with either a program rule or a hypothesis. First suppose that $L_i$ is an atom; $A \leftarrow B_1, \dots, B_k$ is a standardized apart variant of a rule of $P$; and $\theta$ is the *mgu* of $L_i$ and $A$. Then the following is an instance of the rule:

$$\frac{\Gamma\,(L_1 \dots L_{i-1},\, L_i,\, L_{i+1} \dots L_n \mid H)\,\Delta}{\left[\Gamma\,(L_1 \dots L_{i-1},\, B_1, \dots, B_k,\, L_{i+1} \dots L_n \mid H)\,\Delta\right]\theta}\,.$$

Next, let $L_i$ be a (possibly negative) literal; let $L'$ be a hypothesis; and let $\theta$ be the *mgu* of $L_i$ and $L'$. Then the following is an instance of the rule:

$$\frac{\Gamma\,(L_1 \dots L_{i-1},\, L_i,\, L_{i+1} \dots L_n \mid H, L')\,\Delta}{\left[\Gamma\,(L_1 \dots L_{i-1},\, L_{i+1} \dots L_n \mid H, L')\,\Delta\right]\theta}\,.$$

## *6.2 Contradiction*

This rule tries to prove $(G \mid H)$ "vacuously," by showing that the hypotheses $H$ cannot be satisfied by any stable model of $P$. Hereafter $\bar{L} = \mathtt{not}\,A$ if $L$ is an atom $A$, and $\bar{L} = A$ if $L = \mathtt{not}\,A$:

$$\frac{\Gamma\,(G \mid H, L)\,\Delta}{\Gamma\,(\bar{L} \mid H, L)\,\Delta}\,.$$

## *6.3 Split*

Essentially, this rule is needed to compute floating conclusions and discover contradictions. It splits the search space by introducing two new, complementary hypotheses. Let $G_0$ be the *restart goal* (i.e., the left-hand side of the first h-goal of the derivation), $L$ be an arbitrary literal and $\sigma$ be the composition of the *mgu*s previously computed during the derivation; the Split rule is

$$\frac{\Gamma\,(G \mid H)\,\Delta}{\Gamma\,(G \mid H, L)\,(G_0\sigma \mid H, \bar{L})\,\Delta}\,.$$

### *6.4 Success*

This is a structural rule that removes h-goals once they have been successfully proved. As usual, $\Box$ denotes the empty goal:

$$\frac{\Gamma\,(\Box \mid H)\,\Delta}{\Gamma\,\Delta}\,.$$

### *6.5 Counter-supports*

We are left to illustrate the last rule of the calculus, that models negation as failure. In order to abstract away the details of the computation of failed facts, the rule is expressed in terms of so-called counter-supports that in turn are derived from the standard notion of *support*. Recall that a support for a ground atom $A$ is a set of negative literals obtained by applying SLD resolution to $A$ with respect to the given program $P$ until no positive literal is left in the current goal. (The final, negative goal of the SLD derivation is a support for $A$.)

*Definition 6.1* (*Bonatti 2001b*)
Let $A$ be a ground atom. A *ground counter-support* for $A$ in a program $P$ is a set of atoms $K$ with the following properties:

1. for each support $S$ for $A$, there exists $\texttt{not}\,B \in S$ such that $B \in K$;
2. for each $B \in K$, there exists a support $S$ for $A$ such that $\texttt{not}\,B \in S$.

In other words, the first property says that $K$ contradicts all possible ways of proving $A$, while the second property is a sort of relevance property. Informally speaking, the failure rule of skeptical resolution says that if all atoms in a counter-support are true, then all attempts to prove $A$ fail, and hence $\texttt{not}\,A$ can be concluded.

Of course, in general, counter-supports are not computable and may be infinite (while skeptical derivations and their goals should be finite).

In Bonatti (2001b) the notion of counter-support is generalized to nonground atoms in the following way:

*Definition 6.2*
A (generalized) *counter-support* for a ground atom $A$ is a pair $\langle K, \theta\rangle$, where $K$ is a set of atoms and $\theta$ a substitution, such that for all grounding substitutions $\sigma$, $K\sigma$ is a ground counter-support for $A\theta\sigma$.

The actual mechanism for computing counter-supports can be abstracted by means of a suitable function $\mathsf{CounterSupp}$, mapping each (possibly nonground) atom $A$ onto a set of *finite* generalized counter-supports for $A$. The underlying intuition is that function $\mathsf{CounterSupp}$ captures all the negative inferences that can actually be computed by the chosen implementation. Now negation-as-failure can be axiomatized as shown next.

### *6.6 Failure*

Suppose that $L_i = \texttt{not}\,A$ and $\langle\{B_1, \dots, B_k\}, \theta\rangle \in \mathsf{CounterSupp}(A)$. Then the following is an instance of the Failure rule:

$$\frac{\Gamma\,(L_1 \dots L_{i-1},\ L_i,\ L_{i+1} \dots L_n \mid H)\,\Delta}{\left[\Gamma\,(L_1 \dots L_{i-1},\ B_1, \dots, B_k,\ L_{i+1} \dots L_n \mid H)\,\Delta\right]\theta}\,.$$

To achieve completeness for the nonground skeptical resolution calculus, we need the negation-as-failure mechanism to be complete in the following sense:

*Definition 6.3*
The function CounterSupp is *complete* iff for each atom $A$, for all of its ground instances $A\gamma$, and for all ground counter-supports $K$ for $A\gamma$, there exist $\langle K', \theta\rangle \in \mathsf{CounterSupp}(A)$ and a substitution $\sigma$ such that $A\theta\sigma = A\gamma$ and $K'\sigma = K$.

## 6.7 *Skeptical derivations*

A *skeptical derivation from P and* CounterSupp *with restart goal* $G_0$ is a (possibly infinite) sequence of s-goals $\Gamma_0, \Gamma_1, \ldots$, where each $\Gamma_{i+1}$ is obtained from $\Gamma_i$ through one of the five rewrite rules of the calculus. A skeptical derivation is *successful* if its last s-goal is empty; in this case we say that the first s-goal has a successful skeptical derivation from $P$.

*Example 6.4*
Let $P$ be

1. $p(X) \leftarrow \mathtt{not}\, q(X)$
2. $q(X) \leftarrow \mathtt{not}\, p(X)$
3. $r(f(X)) \leftarrow \mathtt{not}\, p(X)$
4. $r(f(X)) \leftarrow \mathtt{not}\, q(X)$

For all ground terms $t$, the literal $\mathtt{not}\, p(t)$ is the unique support of $q(t)$. Therefore, we can set $\mathsf{CounterSupp}(q(X)) = \{\langle p(X), \varepsilon\rangle\}$ (where $\varepsilon$ denotes the empty substitution), since the truth of $p(X)$ suffices to block all derivations of $q(X)$, for all possible values of $X$ (the issue of how to compute CounterSupp will be briefly discussed at the end of this section). The following is a successful derivation of $(r(Y) \mid \emptyset)$ from $P$ with answer substitution $[Y/f(X)]$, showing that for all $X$, $r(f(X))$ is a skeptical consequence of $P$.

$$\begin{array}{rl}
(r(Y) \mid \emptyset) & \\
(\mathtt{not}\, p(X) \mid \emptyset) & \text{by resolution with 3; it binds } Y \text{ to } f(X); \\
(\mathtt{not}\, p(X) \mid \mathtt{not}\, p(X))\,(r(f(X)) \mid p(X)) & \text{by the splitting rule;} \\
(\Box \mid \mathtt{not}\, p(X))\,(r(f(X)) \mid p(X)) & \text{by resolution with the hypothesis;} \\
(r(f(X)) \mid p(X)) & \text{by the success rule;} \\
(\mathtt{not}\, q(X) \mid p(X)) & \text{by resolution with 4;} \\
(p(X) \mid p(X)) & \text{by the failure rule using } \langle p(X), \varepsilon\rangle; \\
(\Box \mid p(X)) & \text{by resolution with the hypothesis;} \\
\Box & \text{by the success rule.}
\end{array}$$

Skeptical resolution is sound for *all* normal programs and counter-support calculation mechanisms, as stated in the following theorem:

*Theorem 6.5* (*Soundness; Bonatti 2001b*)
Suppose that an *s-goal* $(G \mid H)$ has a successful skeptical derivation from a normal program $P$ and CounterSupp with restart goal $G$ and answer substitution $\theta$. Then, for all grounding substitution $\sigma$, all the stable models of $P$ satisfy $(\bigwedge G\theta \leftarrow \bigwedge H\theta)\sigma$ (equivalently, $\forall(\bigwedge G\theta \leftarrow \bigwedge H\theta)$ is skeptically entailed by $P$).

However, skeptical resolution is not always complete. Completeness analysis is founded on ground skeptical derivations that require a ground version of $\mathsf{CounterSupp}$.

*Definition 6.6*
For all ground atoms $A$, let $\mathsf{CounterSupp}^g(A)$ be the least set such that if $\langle K, \theta\rangle \in \mathsf{CounterSupp}(A')$ and for some grounding $\sigma$, $A = A'\theta\sigma$, then

$$\langle K\sigma, \epsilon\rangle \in \mathsf{CounterSupp}^g(A),$$

where $\epsilon$ is the empty substitution.

*Theorem 6.7* (*Finite Ground Completeness, Bonatti 2001b*)
If some ground implication $\bigwedge G \leftarrow \bigwedge H$ is skeptically entailed by a *finite* ground program $P$ and $\mathsf{CounterSupp}$ is complete with respect to $P$, then $(G \mid H)$ has a successful skeptical derivation from $P$ and $\mathsf{CounterSupp}^g$ with restart goal $G$. In particular, if $G$ is skeptically entailed by $P$, then $(G \mid \emptyset)$ has such a derivation.

This basic theorem and the following standard lifting lemma allow to prove completeness for all finitely recursive normal programs.

*Lemma 6.8* (*Lifting; Bonatti 2001b*)
Let $\mathsf{CounterSupp}$ be complete. For all skeptical derivations $\mathscr{D}$ from a normal program $P$ and $\mathsf{CounterSupp}^g$ with restart goal $G_0$, there exists a substitution $\sigma$ and a skeptical derivation $\mathscr{D}'$ from $P$ and $\mathsf{CounterSupp}$ with restart goal $G_0'$ and answer substitution $\theta$, such that $\mathscr{D} = \mathscr{D}'\theta\sigma$ and $G_0 = G_0'\theta\sigma$.

*Theorem 6.9* (*Completeness for finitely recursive normal programs*)
Let $P$ be a finitely recursive normal program. Suppose $\mathsf{CounterSupp}$ is complete with respect to $P$ and that for some grounding substitution $\gamma$, $(\bigwedge G \leftarrow \bigwedge H)\gamma$ holds in all the stable models of $P$. Then $(G \mid H)$ has a successful skeptical derivation from $P$ and $\mathsf{CounterSupp}$ with restart goal $G$ and some answer substitution $\theta$ more general than $\gamma$.

*Proof*
By Theorems 3.4 and 5.2, there exists a smooth module sequence for $P$ with finite elements $P_1$, $P_2$, …, and a finite $k$ such that $(\bigwedge G \leftarrow \bigwedge H)\gamma$ holds in all the stable models of $P_k$. Since each $P_i$ is downward closed, the ground supports of any given $A \in atom(P_k)$ with respect to program $P_k$ coincide with the ground supports of $A$ with respect to the entire program $P$. Consequently, also ground counter-supports and (generalized) counter-supports, respectively, coincide in $P_k$ and $P$. Therefore, $\mathsf{CounterSupp}$ is complete with respect to $P_k$, too. As a consequence, since $P_k$ is a ground, finite program, the ground completeness theorem can be applied to conclude that $(G \mid H)\gamma$ has a successful skeptical derivation from $P_k$ and $\mathsf{CounterSupp}^g$ with restart goal $G\gamma$. The same derivation is also a derivation from $P$ (as $P_k \subseteq \mathsf{Ground}(P)$) and $\mathsf{CounterSupp}^g$. Then, by the Lifting lemma (note that $P$ is supposed to be normal), $(G \mid H)$ has a successful skeptical derivation from $P$ and $\mathsf{CounterSupp}$, with restart goal $G$ and some answer substitution $\theta$, such that $(G \mid H)\gamma$ is an instance of $(G \mid H)\theta$. It follows that $\theta$ is more general than $\gamma$. □

An important question is whether any computable, complete function $\mathsf{CounterSupp}$ exists. Take any module sequence $P_1, P_2, \ldots, P_i, \ldots$ based on any effective enumeration of $GH$. Note that for all ground atoms $A$ one can effectively find a $k \in \mathbb{N}$ such that $A \in$

$atom(P_k)$. Now, if the given program $P$ is finitely recursive, then the ground supports of $A$ can be computed by building all the acyclic SLD derivations for $A$, using the finitely many ground rules of $P_k$. Consequently, the ground counter-supports of $A$ are finite and finitely many, too, and can be easily computed from the ground supports of $A$. Now consider a nonground atom $A$. Let $\mathsf{CounterSupp}(A)$ be the set of pairs $\langle K, \gamma \rangle$ such that $\gamma$ is a grounding substitution for $A$ and $K$ is a ground counter-support for $A\gamma$. Clearly, for any given $A$ such pairs can be recursively enumerated by enumerating the ground instances of $A$ and computing for each of them the corresponding ground counter-supports as explained above. Clearly, this counter-support function is complete by construction. This proves the following:

*Theorem 6.10*
If $P$ is normal and finitely recursive, then there exists a complete $\mathsf{CounterSupp}$ function such that for all atoms $A$, $\mathsf{CounterSupp}(A)$ is recursively enumerable.

This property allows to recursively enumerate all skeptical derivations from $P$. Therefore, skeptical resolution provides an alternative proof that skeptical inference from finitely recursive normal programs is in r.e.

## 7 Finitary programs and other decidable fragments

The inherent complexity of finitely recursive programs calls for further restrictions to make deduction decidable.

One of such additional restrictions is based on the following idea: Suppose that there exists a module sequence $P_1, P_2, \ldots, P_i, \ldots$ and an index $k$ such that for all interpretations $I \subseteq atom(P_k)$, the "top" program $e_{atom(P_k)}(\mathsf{Ground}(P) \setminus P_k, I)$ is consistent. Then the splitting theorem guarantees that every stable model of $P_k$ can be extended to a stable model of $P$, and, conversely, every stable model of $P$ extends a stable model of $P_k$. As a consequence, given a ground goal $G$ (be it credulous or skeptical) whose atoms are included in $atom(P_k)$, the answer to $G$ can be computed by inspecting only the stable models $M_{k,1}, \ldots, M_{k,n}$ of $P_k$ (which is a finite ground program if $P$ is finitely recursive). The "upper" part of the stable models of $P$, that is, the stable models of $e_{atom}(P_k)(\mathsf{Ground}(P) \setminus P_k, M_{k,i})$ $(1 \leqslant i \leqslant n)$, need not be computed at all—we only need to know that they exist to be confident that $M_{k,1}, \ldots, M_{k,n}$ are sufficient to answer $G$.

This is the idea underlying *finitary programs* (Bonatti 2004). For normal programs, the consistency of the top program is guaranteed by means of a theorem due to Fages (1994), stating that *order consistent* normal programs are always consistent. A normal program is order consistent if there exists no infinite sequence of (possibly repeated) atoms $\langle A_i \rangle_{i<\omega}$ such that $A_i$ depends both positively and negatively on $A_{i+1}$ for all $i < \omega$. For example, all positive programs are trivially order consistent, while Fages's (1994) program

$$q(X) \leftarrow q(f(X))$$
$$q(X) \leftarrow \texttt{not}\, q(f(X))$$

exploited in Example 3.10 is not, as well as any normal program whose dependency graph contains some odd cycle. The above program shows that a program may fail to be order consistent even if the program is acyclic. However, if $P$ is normal and finitely recursive, then it can be shown that $P$ is order consistent iff $P$ is odd-cycle-free (Bonatti 2004). This observation justifies the definition of finitary programs (Definition 2.2): by requiring finitary programs to have finitely many odd cycles, it is possible to confine all odd cycles into a single, finite program module $P_k$ and ensure that the "top" programs are odd-cycle-free and hence consistent.

As proved in (Bonatti 2004), the extra condition on odd cycles suffices to make both credulous and skeptical ground queries decidable. However, in Bonatti (2004) the statement erroneously fails to include the set of odd-cyclic literals among the inputs of the algorithm. The correct statement and a slightly different proof based on module sequences is given next.

*Theorem 7.1*
Given a finitary normal program $P$ and a finite set $C$ containing (at least) all of the odd-cyclic atoms of $P$'s Herbrand base,

(i) whether a ground formula $G$ is a credulous consequence of $P$ is decidable;
(ii) whether a ground formula $G$ is a skeptical consequence of $P$ is decidable.

*Proof*
(Sketch) Let $P_1, P_2, \ldots, P_i, \ldots$ be any (recursive) module sequence induced by a recursive enumeration of $P$'s Herbrand base, and let $k$ be the minimal index such that $C \cup atom(G) \subseteq atom(P_k)$. Clearly, such a $k$ exists and is effectively computable. Moreover, $P_k$ is ground and finite (because $P$ is finitely recursive); therefore the set of its stable models $M_{k,1}, \ldots, M_{k,n}$ can be effectively computed as well; it is finite, and consists of finite models. Now, by construction, the "top" programs $e_{atom(P_k)}(\mathsf{Ground}(P) \setminus P_k, M_{k,i})$ $(1 \leqslant i \leqslant n)$ are all odd-cycle-free—and hence consistent, by Fages's (1994) theorem. It follows by the splitting theorem that for all $i = 1, \ldots, n$, the program $P$ has a stable model $M$ such that $M \cap atom(P_k) = M_{k,i}$. As a consequence, if $G$ is true (respectively false) in a stable model of $P_k$, then $G$ must be true (respectively false) in a stable model of $P$. Conversely, by the splitting theorem, if $G$ is true (respectively false) in a stable model of $P$, then $G$ must be true (respectively false) in a stable model of $P_k$ (because $atom(P_k)$ splits $P$). It follows easily that $G$ is a credulous (respectively skeptical) consequence of $P$ iff $G$ is a credulous (respectively skeptical) consequence of $P_k$. Of course, since the set of stable models of $P_k$ is finite, recursive, and contains only finite models, both the credulous and the skeptical consequences of $P_k$ are decidable. $\square$

Extending this result to disjunctive programs is not a trivial task because, unfortunately, Fages's (1994) theorem does not scale to disjunctive programs in any obvious way. Consider the possible natural generalization of atom dependencies from the class of normal programs to the class of disjunctive programs:

1. First assume that the unlabeled edges of $DG(P)$ are ignored; that is, let $A$ depend on $B$ iff there is a path from $A$ to $B$ in $DG(P)$ with no unlabeled edges. This is equivalent to adopting a dependency graph similar to the traditional graphs for normal programs, with no head-to-head edges. Using the resulting notion of atom dependencies, one can find programs that are order consistent but have no stable models. One of them

is

$$p_1 \vee p_2$$
$$q_1 \vee q_2$$
$$p_1 \leftarrow \texttt{not}\, q_1$$
$$q_1 \leftarrow \texttt{not}\, p_2$$
$$p_2 \leftarrow \texttt{not}\, q_2$$
$$q_2 \leftarrow \texttt{not}\, p_1\,.$$

2. Next, suppose that unlabeled edges are regarded as positive edges; that is, $A$ depends positively (respectively negatively) on $B$ iff there is a path from $A$ to $B$ in $DG(P)$ with an even (respectively odd) number of negative edges. The above inconsistent program is still order consistent under this new notion of dependency.
3. Finally, assume that unlabeled edges are regarded as negative edges. This is a natural assumption given the minimization-based nature of disjunctive stable models: For instance, if $P = \{p \vee q\}$, then the falsity of $p$ implies the truth of $q$ and vice versa. (Indeed $P$ is equivalent to $\{p \leftarrow \texttt{not}\, q,\ q \leftarrow \texttt{not}\, p\}$.) A major problem is that with this form of dependency, too many interesting disjunctive programs are *not* order consistent:
   - every rule with at least three atoms in the head generates an odd cycle through those atoms; therefore the program would not be order consistent;
   - for every cycle $\mathscr{C}$ containing a head-to-body edge $(A, \pm, B)$ originated by a "proper" disjunctive rule (i.e., a rule with two or more atoms in the head) there exists an odd cycle (possibly $\mathscr{C}$ itself or the cycle obtained by extending $\mathscr{C}$ with a negative edge from $A$ to another atom in the same head). This means that disjunctive rules could never be applied in any recursion.

Similar problems (preconditions that are difficult to ensure in practical cases) affect Turner's (1994) approach to consistency. His *signed programs* generalize order consistent normal programs as follows: It should be possible to partition the Herbrand base into two sets $H_1$ and $H_2$ such that

1. negative edges always cross the two partitions; positive edges never do;
2. each rule head is entirely contained in a single partition;
3. the set of rules whose head is contained in $H_1$ is a normal program.

Unfortunately, to the best of our knowledge no application domains naturally require programs satisfying the third condition (that roughly speaking makes the program "half normal").

A more recent paper (Bonatti 2002) ensures consistency through the theory of program *shifting* (Bonatti 1993). A shifting of $P$ is a modified version of $P$ in which some atoms are moved from heads to bodies and enclosed in the scope of a negation symbol. This transformation preserves the classical semantics of the program but not its stable models. However, every stable model of a shifted program is also a stable model of the original program, so the consistency of the former implies the consistency of the latter. Then the approach of Bonatti (2002) consists in adding more conditions to the definition of finitary programs to ensure that at least one "full" shifting of $P$—transforming $P$ into a normal program—is finitary, so that the original consistency theorem by Fages (1994) can be applied. The main drawback of this approach is that the extra conditions required are clumsy and—again—difficult to use in practice.

A very interesting and novel recent approach by Simkus and Eiter (2007) consists in replacing the consistency property with other properties enjoyed by some decidable fragments of first-order logic such as description logics and the guarded fragment. In these fragments, consistent theories always have both a finite model and a tree model which is the "unwinding" of the finite model, i.e., a regular tree. Syntactic restrictions on predicate arity and on the occurrences of function symbols (modeled around the skolemization of guarded formulae) have been exploited to prove the decidability of a new class of finitely recursive programs called FDNC programs. In our framework, this idea roughly corresponds to having regular module sequences in which after some steps the new rules contained in $P_i \setminus P_{i-1}$ are isomorphic to some previous program slice $P_j \setminus P_{j-1}$ ($j < i$). Therefore in order to find a stable model of $P$ one needs only to find a stable model $M$ for some finite module $P_i$, as a model for the upper part can then be constructed by cloning $M$ or submodels thereof. FDNC programs can be applied to encode ontologies expressed in description logics and are suitable to model a wide class of planning problems. An interesting open question is whether this approach can be generalized to wider interesting classes of programs by studying regular module sequences.

## 8 Conclusions

In this paper we have extensively studied the properties of stable model reasoning with disjunctive, finitely recursive programs—a very expressive formalism for ASP. Finitely recursive programs extend the class of finitary programs by dropping the restrictions on odd cycles, that is, on the number of possible sources of inconsistencies. We extended to finitely recursive programs many of the nice properties of finitary programs: (i) a compactness property (Theorem 4.5); (ii) the r.e.-completeness of inconsistency checking and skeptical inference (Theorem 5.4); (iii) the completeness of skeptical resolution (Theorem 6.9); note that this result applies to normal programs only, unlike (i) and (ii).

Unfortunately, some of the nice properties of finitary programs do *not* carry over to finitely recursive programs: (i) ground queries are not decidable (Theorem 5.5 and Corollary 5.6); (ii) nonground credulous queries are not semidecidable (Corollary 5.6).

We proved our results by extending the splitting sequence theorem that, in general, guarantees only that each consistent program $P$ has a consistent module sequence for $P$. We proved that in general the converse does not hold (Example 3.10), unless $P$ is finitely recursive: in that case, the stable models of a consistent module sequence always converge to a model of $P$ (Theorem 4.1).

As a side benefit, our techniques introduce a normal form for splitting sequences and their bottom programs, where sequence length is limited to $\omega$ and—if the program is finitely recursive—the sequence is smooth (i.e., the "delta" between each non-limit element and its predecessor is finite). Such properties constitute an alternative characterization of finitely recursive programs. The theory of module sequences is a powerful tool for working on decidable inference with infinite stable models, as it provides a constructive, iterative characterization of the stable models of a large class of programs with infinite domains. In Section 7 we carried out a first attempt at relating different approaches using module sequences as a unifying framework. However such an analysis is still very preliminary and partially informal; its development constitutes an interesting subject for future work, and it may contribute to recent areas such as research on FDNC programs.

Another interesting open problem is extending to disjunctive programs Fages's (1994) consistency result (an important ingredient in several decidability results). The existing approaches are based on rather restrictive assumptions that call for more flexible solutions.

Finally, an interesting theoretical question is whether skeptical resolution can be extended to disjunctive programs. A related challenge is finding a satisfactory goal-directed calculus for the positive fragment, which is based on a minimal model semantics.

## *Acknowledgements*

This work was partially supported by the PRIN project Enhancement and Applications of Disjunctive Logic Programming, funded by the Italian Ministry of Research (MIUR).

## References

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge.

Baselice, S., Bonatti, P. A. and Criscuolo, G. 2007. On finitely recursive programs. In *ICLP*, V. Dahl and I. Niemelä, Eds. Lecture Notes in Computer Science, vol. 4670. Springer, 89–103.

Bonatti, P. A. 1993. *Shift-Based Semantics: General Results and Applications. Tech. Rep. CD-TR 93/59*, Computer Science Department, Institute of Information Systems, Technical University of Vienna, Vienna.

Bonatti, P. 2001a. Prototypes for reasoning with infinite stable models and function symbols. In *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001*, T. Eiter, W. Faber, and M. Truszczyǹski, Eds. Lecture Notes in Computer Science, vol. 2173. Springer, Vienna, Austria, 416–419.

Bonatti, P. A. 2001b. Resolution for skeptical stable model semantics. *Journal of Automatic Reasoning 27*, 4, 391–421.

Bonatti, P. A. 2002. Reasoning with infinite stable models II: Disjunctive programs. In *ICLP*, P. J. Stuckey, Ed. Lecture Notes in Computer Science, vol. 2401. Springer, 333–346.

Bonatti, P. A. 2004. Reasoning with infinite stable models. *Artificial Intelligence 156*, 1, 75–111.

Calimeri, F., Cozza, S., Ianni, G. and Leone, N. 2008. Computable functions in ASP: Theory and implementation. In *ICLP*, M. G. de la Banda and E. Pontelli, Eds. Lecture Notes in Computer Science, vol. 5366. Springer, 407–424.

Eiter, T., Leone, N., Mateis, C., Pfeifer, G. and Scarcello, F. 1997. A deductive system for non-monotonic reasoning. In *Logic Programming and Nonmonotonic Reasoning, 4th International Conference, LPNMR'97, Proceedings*,

J. Dix, U. Furbach, and A. Nerode, Eds. Lecture Notes in Computer Science, vol. 1265. Springer, Dagstuhl, Germany, 364–375.

FAGES, F. 1994. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science 1*, 51–60.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *Proc. of the 5th ICLP*. MIT Press, 1070–1080.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9,* 3–4, 365–386.

LIFSCHITZ, V. AND TURNER, H. 1994. Splitting a logic program. In *International Conference on Logic Programming*, P. Van Entenryck, Ed. S. Margherita Ligure, MIT Press, 23–37.

LLOYD, J. W. 1984. *Foundations of Logic Programming*, 1st ed. Springer.

MAREK, V. AND REMMEL, J. 2001. On the expressibility of stable logic programming. In *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001*, T. Eiter, W. Faber, and M. Truszczyǹski, Eds. Lecture Notes in Computer Science, vol. 2173. Springer, Vienna, Austria, 107–120.

MAREK, V. W. AND TRUSZCZYNSKI, M. 1998. Stable models and an alternative logic programming paradigm. Computing Research Repository, CoRR cs.LO/9809032. (http://arxiv.org/corr)

NIEMELÄ, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3–4, 241–273.

NIEMELÄ, I. AND SIMONS, P. 1997. Smodels—an implementation of the stable model and well-founded semantics for normal LP. In *Logic Programming and Nonmonotonic Reasoning, 4th International Conference, LPNMR'97, Proceedings*, J. Dix, U. Furbach, and A. Nerode, Eds. Lecture Notes in Computer Science, vol. 1265. Springer, Dagstuhl, Germany, 421–430.

SIMKUS, M. AND EITER, T. 2007. FDNC: Decidable non-monotonic disjunctive logic programs with function symbols. In *14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2007*, N. Dershowitz and A. Voronkov, Eds. Lecture Notes in Computer Science, vol. 4790. Springer, Yerevan, Armenia, 514–530.

TURNER, H. 1994. Signed logic programs. In *SLP*, M. Bruynooghe, Ed. MIT Press, Itaha, New York, 61–75.

TURNER, H. 1996. Splitting a default theory. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, H. Shrobe and T. Senator, Eds. AAAI Press, Menlo Park, CA, 645–651.